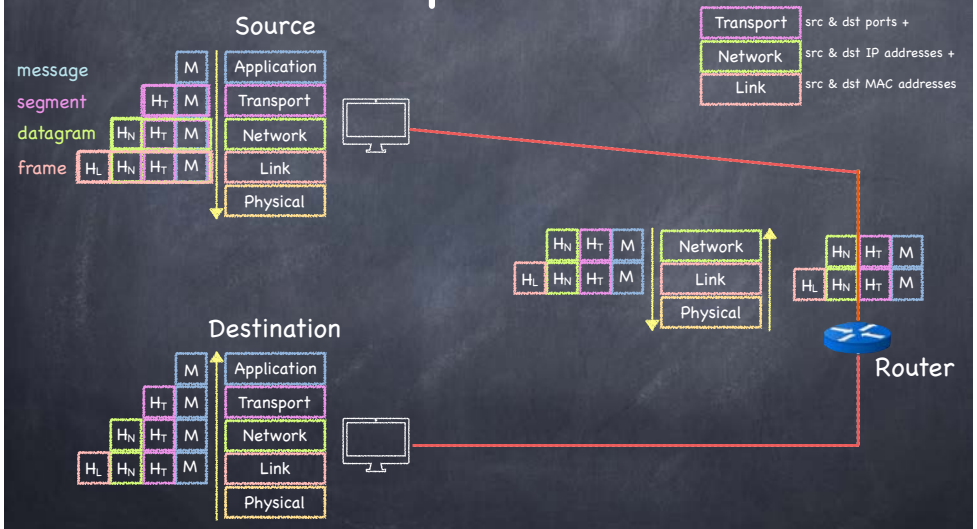
Encapsulation



End-to-End Argument

Saltzer, Reed & Clark, 1981

- If a function can be completely and correctly implemented only with the knowledge and help of the application standing at the endpoints of the communication system,
- then providing that function as a feature of the communication system itself is not possible
- Sometimes providing an incomplete version as a feature of the communication system itself may be useful as a performance enhancement

An Application of the Argument

- Should the network guarantee packet delivery?
  - Consider transferring a file
    - Sender reads file from disk & sends it; Receiver reads packets and writes them to disk
  - Wouldn't it be simpler if network guaranteed delivery?
  - No!
    - Application still needs to check file was written to disk
    - It needs to implement anyway its own retransmits

EtE Argument's Impact

- Occam's Razor for Internet architecture
  - End-to-end properties are best provided by applications, not by the network
    - Guaranteed packet delivery, ordered packet delivery, duplicate suppression, security, etc.
- Internet performs simplest packet routing and delivery service
  - Packets are sent on a best-effort basis

Application Layer

Application Layer Protocols

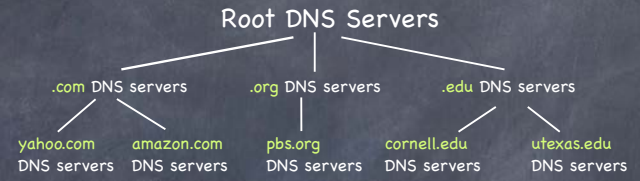
- HTTP
  - persistent/non persistent; stateless, hence cookies
- Electronic Mail
  - SMTP; POP3; IMAP
- DNS
  - Manages naming of internet hosts

Naming

- People
  - SSN, NetId, Passport Number
- Internet Hosts, Routers
  - IP Address (32 bit) Ex: 128.84.96.12
    - Assigned to hosts by their internet service providers (ISPs)
    - Not a unique id, can be reused for a different machine
    - Determines how packets reach its holder
  - A virtual "name" Ex: www.cs.cornell.edu
    - Human friendly

How are human friendly names translated into IP addresses?

Domain Name System (DNS) Mockapetris '87

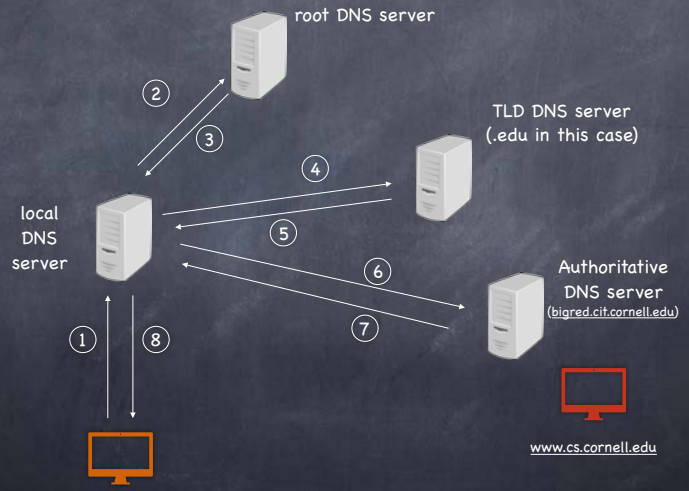


- A name service built above a hierarchical, distributed, autonomous, reliable database
  - Application level protocol: hosts & name servers communicate to resolve names
  - Introduced to replace the original Internet naming scheme
    - a single central master file downloaded everywhere by FTP
  - Components separated by dot and resolved from right to left
  - All names are global: they mean the same everywhere in the DNS

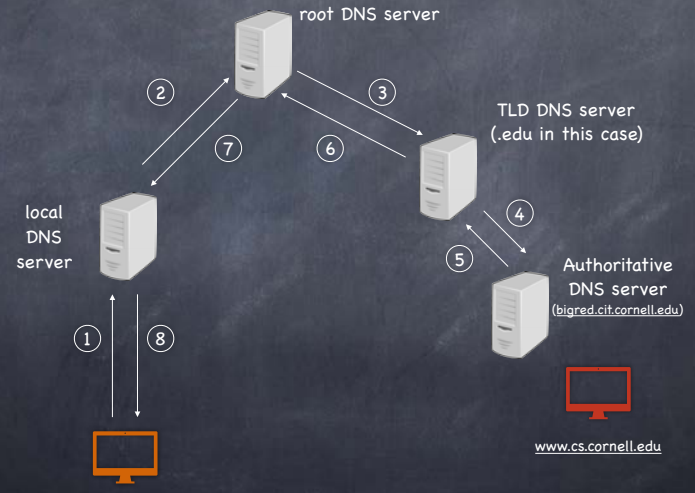
DNS Lookup

- To find the IP address of www.cs.cornell.edu (basic protocol)
  - Query root server to find IP of DNS server for edu...
  - which, when queried next, will provide IP of DNS server for cornell.edu
  - which, when queried next, will provide IP of cs.cornell.edu
  - which, when queried next, will provide IP of www.cs.cornell.edu

DNS Name Resolution: Iterative



DNS Name Resolution: Recursive



Caching

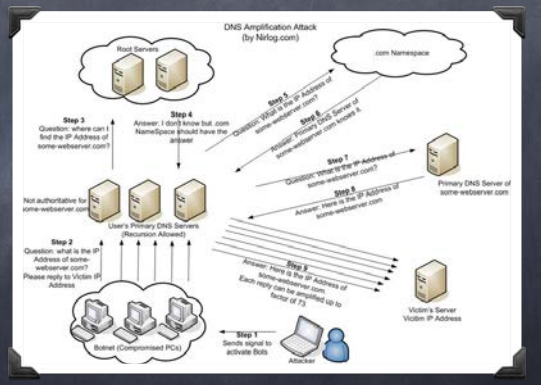
- Cache entries may be out of date
  - bindings are forgotten after TTL
  - TLD servers typically cached in local name servers
- Servers cache new bindings they learn
  - only eventual consistency
  - if binding changes, it may not be learned until TTL
  - update notify proposed IETF standard

Attacking DNS

- DDos attacks on DNS
  - target: root servers
    - to date, unsuccessful
    - traffic filtering / root server bypass via TLD, cashed at local servers
  - target: TLD servers
    - potentially more dangerous
- Redirect attacks
  - Man-in-the-middle
  - DNS poisoning

Leveraging DNS for DDoS

- DDOS Amplification
  - asymmetric attack
  - send query with spoofed source address
    - IP of the victim



Remote Procedure Calls

The Client Server Paradigm

- Server: Program(s) that provide some service
- Client: Program that uses the service
- Typical pattern:
  - Client connects to the server
    - locates it in the network and establishes a connection
  - Client sends requests
    - messages that indicate desired service and necessary parameters
  - Server returns response

Pros and Cons of Messages

- Very flexible communication
  - Programmers can choose message's format as they see fit
- But...
  - Now programmer must worry about message formats
  - Messages must be packed and unpacked
  - Messages must be decoded by server to determine requested service
  - Messages may require special error handling functions

Procedure Calls to the Rescue

- A more natural way to communicate
  - supported by every language
  - with well-defined semantics
- The idea
  - Server exports a set of procedures, that clients can invoke, as if they were co-located
- The rub
  - They are not colocated!
    - How do we make this transparent to the programmer?

Remote Procedure Call (RPC)

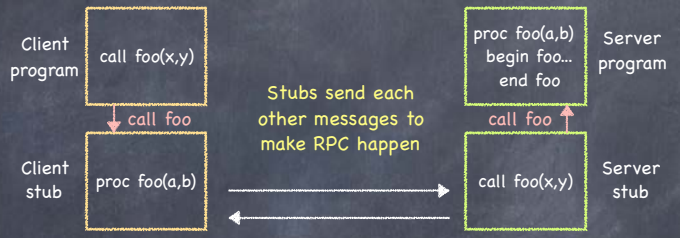
Birrell & Nelson @ Xerox PARC
"Implementing Remote Procedure Calls" (1984)

- Goal: design RPC to look like a local procedure call
  - Client and server each implement three components
    - user program
    - a set of stop procedures
    - RPC runtime support

Building a Server

- Interface defined using an Interface Definition Language (IDL)
  - specifies the names, parameters, and types for all server procedures that clients can call
- Stub compiler reads IDL and produces two stub procedures for each server procedure
  - stubs manage all details of communication between client and server
  - server-side stub is linked to server's code
  - client-side stub is linked to client's

RPC Stubs



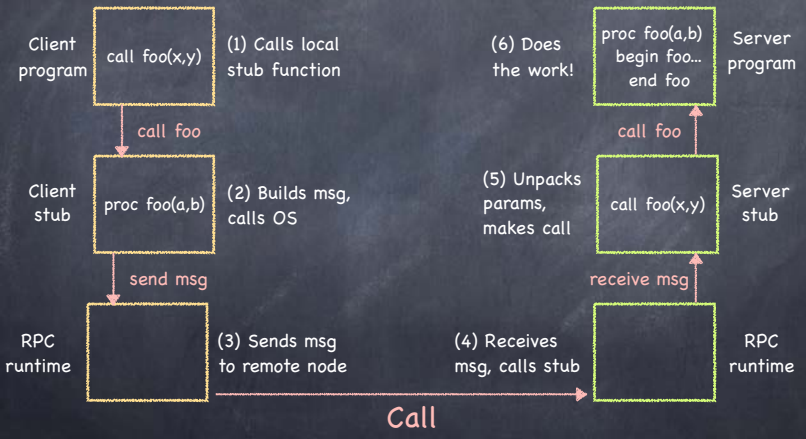
Client-side stub:

- Looks (to client) like a callable server procedure
- Client program thinks it is calling the server

Server-side stub:

- Server program thinks it is called by the client
- foo actually called by the server stub

RPC Call Structure



RPC Return Structure